Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 96 (2016) 1361 – 1370

Procedia Computer Science

20th International Conference on Knowledge Based and Intelligent Information and Engineering Systems, KES2016, 5-7 September 2016, York, United Kingdom

Multi-stage monitoring of abnormal situation based on complex event processing

Tao Lu*, Xinxin Zha, Xin Zhao

Institute of system engineering,Dalian University of Technology,Dalian,China

Abstract

This paper focuses on the monitoring of abnormal situation in workspace where complicate production activities are performed and possible abnormal situations vary in different stages. The monitoring application should track the production process, identifying the production stage and detecting anomaly in every stage as defined. With the development of ubiquitous computing technology and widespread of sensing equipment, context information pertaining to smart working environment is available for monitoring applications. Complex event processing (CEP) is usually introduced to process and correlate context information for its attractive feature of extracting composite event from a large amount of event data in real time according to user-defined event patterns. In this paper, we present context model and event model in which discrete event such as acquiring context value at a point of time is represented by context. The abnormal situation in every stage of production can be transformed into event expressions, called abnormal event patterns. Contexts in different time captured by sensors form data streams and processed by CEP engine to detect abnormal situation. We propose to use state transition to model each stage so that the normal transition period in the beginning and end of stage can be distinguished from abnormal situation. Once a stage is identified to be starting or ending, the application will change abnormal event patterns accordingly. Case study about metallographic examination proves that the approach we propose is effective and feasible for some multi-stage abnormal situation monitoring.

Peer-review under responsibility of KES International

Keywords: complex event processing;context awareness;process monitor

* Corresponding author.
E-mail address: lutao@dlut.edu.cn

Peer-review under responsibility of KES International
doi:10.1016/j.procs.2016.08.181

1. Introduction

Monitoring of abnormal situation in workspace is of great significance in production site management. The expectation of customer for increased safety, product quality, availability and agility, together with the company's need to reduce prices and shorten production times are ongoing challenges for manufacturing enterprises[1]. This gives rise to requirement for advanced monitoring and supervision systems which can timely detect abnormal situation in workspace and send notification to the components in charge of reacting to them.

The advent of ubiquitous computing technologies and widespread of sensors, localization equipment, embedded devices and RFID tags has opened new opportunities to monitor real-world phenomena[2]. Context pertaining to working environment can be captured by embedded sensors and devices. However, the semantic information inside these context data directly acquired from sensors is quite limited. In real application, these low level contexts should be aggregated and properly interpreted, and transformed into high level contexts reflecting normal or abnormal situation of business process[3][4]. For example, if one operation requires two machine be kept on in production process and two machine be turned off within one minutes when the operation is ending, the anomaly that one machine is turned off falsely in operation should be detected by combining the contexts of two machine in a period of time.

Complex event processing (CEP) is usually introduced to process and correlate these context data. The CEP model, which was designed specifically to timely process large amount of flowing data, has gained increasing attention by the academia and industry[5]. CEP can extract composite event from a large amount of basic event according to user- defined event patterns, which are formed by logical and temporal combinations of events from many sources. Compared with delay-analysis method used traditionally in relational database, CEP involves continuous process and analysis of high-volume and high-speed data stream[6]. Many studies explore the use of CEP in ubiquitous computing environment[6][7][8][9][10][11]. For example, Ref.[6] uses a RFID-enabled CEP framework to model complex events in smart hospital. Ref.[7] and Ref.[8] use CEP as a means to create complex events from RFID data stream that can then be used as a foundation for business logic. Ref. [9] provides a formal complex event modeling language to tailor the monitoring task logic by writing simple scripts and then CEP is used for complex event filtering and reasoning to detect exceptions in manufacturing process.

Abnormal situation in workspace can be represented by composite event. However, production activities are mostly complicate and abnormal situations may vary in different stages. Anomaly in one stage may be normal in another stage. Thus the abnormal event patterns to be detected should be changed in different stage. Therefore the monitoring application should track the production process and identify the stage at the same time. We assume that for every stage there exist contexts marking the starting and ending of the stage. When these contexts are sensed, it can be determined that the stage starts or ends. But when the marking contexts can't hold at the same time, e.g. two machines can't be turned off simultaneously, normal transition period of a stage need to be distinguished from abnormal situation.

In this paper, we present context model and event model in which discrete event such as acquiring context at a point of time is represented by context. We propose to use state transition to model each stage of the production process. Context data captured by the sensor is timely processed by the CEP. By this means, the production processed is monitored. Stage transition is identified and abnormal situation can be detected.

The rest of the paper is organized as follows. Section 2 presents context model and event model. Stage division and state transition system are proposed in section 3. Section 4 reports our case study and section 5 concludes the paper.

2. Context model and event Model

Context is information that can be used to characterize the situation of an entity[12]. Context is captured by sensors and embedded devices in smart environment and preprocessed by the middleware to remove noisy data and transformed into standard format which can be understood and further processed by applications. The context is continuous but application acquiring context information is discrete event. Here the event model is closely related with context model.

2.1. Context model

Suppose the application is related to a set of context attributes which are denoted as $A_1, A_2, ...A_n$, and the domain of A_i is D_i. At given time t, the value of context attribute A_i is denoted as $A_i(t)$ $(A_i(t) \in D_i), i = 1,2,..n$.

Suppose $d_j^{(i)} \subseteq D_i$, $c_j^{(i)} = (A_i, d_j^{(i)})$ is called an atomic context which represents situation of an entity or environment. Composite context can be derived from one or more atomic context using conjunction, disjunction and negative operation on atomic contexts. Here conjunction, disjunction and negative operation have their usual meanings.

The context $c_j^{(i)} = (A_i, d_j^{(i)})$ holds at time t, if and only if $A_i(t) \in d_j^{(i)}$, denoted as $hold\_at(c_j^{(i)}, t)$. Apparently $hold\_at((A_i, d_j^{(i)}), t) \Leftrightarrow \neg hold\_at((A_i, D_i - d_j^{(i)}), t)$, $(A_i, D_i - d_j^{(i)})$ is called the negative context of $c_j^{(i)}$, denoted as $\neg c_j^{(i)}$

2.2. Event model

Generally, most of contexts keeping holding over a period of time once it holds. However, the event of acquiring context data from sensors and context processing middleware is instantaneous atomic occurrence at a point of time. Assuming that every context change of interest can be detected and time lag could be omitted, the first time of context c being detected can be regarded as the time of c starting to hold.

Definition 1 (Basic event type). A basic event type can be denoted as $E=e(c)$ where c is an atomic context. An instance of E can be denoted as $e=(e(c), t)$ which means the event that context c is being sensed occurs at time t.

A smart environment may generate a variety of basic event streams. A context c may be sensed several times, and $e(c)$ occurs every time it is sensed.

For example, there is an event stream

$$((m_1, off), 11{:}04), ((m_2, off), 11{:}06), ((m_1, off), 11{:}09), ((m_2, on), 11{:}10), ((m_2, on), 11{:}11), ((m_1, on), 11{:}12)$$

where m_1 and m_2 are two context attributes representing two machine state respectively.

We can see from the event stream that the two machines are turned on in 11:10 and 11:12. The state of m_1 is sensed and reported in 11:04 and 11: 09 before it is turned on.

Definition 2 (Same source event). Suppose $E_1=e(c_j^{(i)}), E_2=e(c_k^{(i)})$, E_1 and E_2 are called same source events.

Same source events are the event related to same context attribute and same source event instances are generated by same sensor at different time.

Since we assume that every context change can be captured in time, an atomic context holding or not at a certain point of time depends on its recent context event or its recent same source event in event stream.

For example, if we want to know the context (m_2, on) holds or not at 11:12, we should check the recent event related to m_2, i.e. $((m_2, on), 11{:}11)$. The context (m_1, off) does not hold at 11:12 though event $e(m_1, off)$ occurs twice in event stream, because the event $e(m_1, on)$, the same source event of $e(m_1, off)$, is the recent event related to m_1 which terminates the context (m_1, off).

Furthermore, composite context at any point of time can also be determined in same way. For above event stream, context $(m_1, off) \wedge (m_2, on)$ holds at 11:10 and $(m_1, on) \wedge (m_2, on)$ holds at 11:12.

Definition 3 (Event type). An event type can be denoted as $E=e(c)$ where c is an atomic context or composite context. If c is a composite context, for an instance of E, the time it occurs is the time last related basic event first time detected.

For example, in above event stream $e((m_1, off) \wedge (m_2, on))$ occurs at 11:10, in which the basic event $e(m_2, on)$

occurs, making the composite context $(m_{1,}\mathit{off}) \wedge (m_{2,}\mathit{on})$ hold for first time.

2.3. Event Pattern

Abnormal situation identification in production site requires detecting abnormal event in real time. Complex Event Processing is suitable for this task. CEP is a software technology for processing high frequent event stream in real time. The basic concept of CEP is in-memory pattern-matching, which means to identify composite event formed by logical and temporal combination of events. The composite events often represent meaningful situation in the application domain, which are defined by users in event computing language.

Composite event is aggregated from basic events and composite event using specific set of event operators such as disjunction, conjunction, sequence and so on, which depends on the specific CEP engine. We choose Esper as a complex event processing tool to test our method proposed in this paper. Esper is open source and widely used, with rich expressive language and extensive documentation[13].

Primitive context and composite context can be identified by CEP. For example, we use Esper sentence below to detect event $e(m_{1,}\mathit{on})$ and $e((m_{1,}\mathit{on}) \wedge (m_{2,}\mathit{on}))$ when they occur.

```
select * from events [every m1(state=On)]

select * from pattern[every ((m1(state='On')-> m2(state='On') and not (m1(state='Off') and m1(state='On'))) or
(m2(state='On')-> m1(state='On') and not (m2(state='Off') and m2(state='On'))))]
```

The “and not” operators are necessary in the second sentence. If event $e(m_{1,}\mathit{on})$, or “m1(state='On')” in the example happens, the “and not” operators cause the subexpression “m1(state='On')-> m2(state='On')” waiting for “m2(state='On')” to end when “m1(state='Off')” or “m1(state='On')” arrives. This ensures that the event matching the pattern is recent same source event.

Composite event is quite different from the composite context event defined in previous section. The composite context event is actually last basic event which make the context hold, and this event may be different in different event stream. The composite event is generally regarded as event with duration, and composed by several basic events which have logical or temporal relations as defined. Though the composite context event can be transformed into composite event to be detected by CEP engine, it is useful by simplifying the expression in the model we proposed in next section.

The possible abnormal situation in production site can be transformed into event expressions, called abnormal event patterns. Abnormal situation is identified in real time by CEP according to the defined abnormal event pattern and actions, such as sending notification, alerting, turning off the power automatically, are taken by the application.

3. Multi-stage monitoring model

Production process may be very complicate and the requirement for abnormal situation monitoring is different and sometimes conflict in different stage. An abnormal situation in one stage may be normal in another one. Therefore it is necessary to divide the production process into stages and monitor the production activity according to different abnormal event patterns in different stages.

3.1 Stage division

Monitoring of different abnormal situation in different stage requires the applications to track the normal production process and identify stage transition at the time of abnormal situation monitoring. Therefore the stage should be distinguishable.

In our proposed method in this paper, for every stage, there should exists a context, atomic or composite, to mark the start (end) of the stage, which is called starting (ending) context. That means when the starting (ending) context begin to hold, the stage is started (ended), and abnormal event patterns for this stage (stage end) is loaded in CEP to

monitor the abnormal situation. Hence the start context should not be abnormal situation in waiting stage and the end context should not be abnormal situation in normal production stage.

Definition 4 (Stage): A stage is a tuple *STAGE*=(*StageId*, C_{start}, C_{end}, P, P_{end}), where *StageId* is stage ID, C_{start}, C_{end}, are start context and end context respectively, P is abnormal event pattern set in this stage, P_{end} is abnormal event pattern set when stage is ended.

P_{end} is necessary for some abnormal event pattern in negative format. This is because for some production activity there are some examinations should be performed at the end. For example, if a detector should be used at least one time in the stage, the event *open detector* and *close detector* having not occurred should not be regarded as abnormal situation in the process of the stage, but is abnormal in the end of the stage and should be checked.

3.2 State transition system

Though there is start context and end context to mark the beginning and ending of the stage, the start and end of the stage should also be tracked and identified. For example, assume that m_1 and m_2 are two machines, both of which should be opened to start a production stage, but any of them is forbidden to be opened in other time. let c_1=(m_1, *on*), c_2=(m_2, *on*), apparently $c_1 \wedge c_2$ is the start context of the stage, $(c_1 \wedge \neg c_2) \vee (\neg c_1 \wedge c_2)$ is abnormal situation. However, in real world, it is impossible for the two context hold at exactly same time, if the machine is opened manually, there could be a transition period that $(c_1 \wedge \neg c_2) \vee (\neg c_1 \wedge c_2)$ holds and the context could be captured by the application. The application should distinguish normal transition from abnormal situation.

The logic to solve this problem proposed in this paper is consistent with the experience in life. That is if it is normal start, context should only be kept holding in short time, otherwise, it is determined that abnormal situation is occurring.

Hence a stage can be furtherly divided into several states: starting states, proceeding states, ending states and abnormal states. Abnormal event patterns are different in different states, and state transition should also be identified. We use state transition system to model this process.

Let $R_{\geq 0}$ be a set of non-negative real numbers.

Definition 5 (State transition system): A state transition system is a tuple $T=(S, \rightarrow, \Sigma, \mu, s_0, S_f)$ where S is a set of states, $\rightarrow \subseteq (S \times \Sigma \times S) \cup (S \times R_{\geq 0} \times S)$ is a transition relation, Σ is a set of event types, $\mu: S \rightarrow R_{\geq 0} \cup \{+\infty\}$ is a function assigning time threshold to every state, s_0 and s_f are initial states and final states respectively.

We write $s \xrightarrow{d} s'$ if $(s,d,s') \in \rightarrow$. If $d \in \Sigma$, $s \xrightarrow{d} s'$ is called event transition, and if $d \in R_{\geq 0}$, $s \xrightarrow{d} s'$ is called time transition.

The state transition system we consider is a variant of timed transition system, here time transitions satisfies part of the standard axioms for delay transitions proposed in Ref[14], such as time additivity and time determinism.

For all $t,t' \in R_{\geq 0}$, $s,s',s'' \in S$:

- **Time additivity**: if $s \xrightarrow{t} s'$ and $s' \xrightarrow{t'} s''$, then $s \xrightarrow{t+t'} s''$
- **Time determinism:** if $t \in R_{\geq 0}$, $s \xrightarrow{t} s'$ and $s \xrightarrow{t} s''$, then $s' = s''$

If $t \in R_{\geq 0}$, $s \xrightarrow{t} s'$, then time determinism ensure that s' is unique, denoted as $s[t]$.

Besides, we require that the system satisfies the following rules:

Rule 1: If $0 \leq t \leq \mu(s)$, then $\exists s' \in S, s' = s[t]$; If $t > \mu(s)$, then $\forall s' \in S, (s,t,s') \notin \rightarrow$

Rule 2: If $0 \leq t \leq \mu(s)$, then $\mu(s[t]) = \mu(s) - t$

Rule 3: $\forall s \in S$, if $\mu(s) = 0$, then $\exists s' \in S, s' \neq s, s \xrightarrow{0} s'$; if $\mu(s) > 0$, then $s \xrightarrow{0} s$

These rules do not satisfy some of standard axioms for delay transition in Ref.[14]. In Ref.[14], transition should not happen without time delay according to axiom **zero delay**. In our proposed model, Rule 3 ensure that this sort of transitions should not happen for the states whose time threshold is not zero, but must happen when time threshold is zero. In addition, though axiom **time continuity** in Ref.[14] is still satisfied in our model, the suitable range is enlarged because Rule 1 ensure that time transition happens whenever $0 \leq t \leq \mu(s)$.

3.3 Modeling stage identification by state transition system

We use state transition system to model the stage identification process. In the model, every transition period such as starting state, ending states are assigned a real-value time threshold to represent maximal delay time. For other states, $+\infty$ are assigned if no time constraints.

Due to the real-value of time threshold, the state-space is infinitely large. However, in the problem we take into account in this paper, the system reacts to the event in every state without considering time difference, i.e., there is a property: If $e \in \Sigma$, $s \xrightarrow{e} s'$, then for $0 < t < \mu(s)$, $s[t] \xrightarrow{e} s'$. Therefore, it can be concluded that system behaviors in state s and $s[t]$ are same when $0 < t < \mu(s)$. In addition, Rule 2 ensures that time constraints is consistent. Hence, state s and $s[t]$ can be regarded as one state in different time, in which transition occurs only when certain event happens or time is up.

Fig.1 presents an example. Suppose c_1 is the starting context, and when c_1 has been holding for 3 minutes, it can be determined that the stage has been started (i.e.$\mu(s_1)$=3m) , otherwise if c_1 terminates holding within 3 minutes, state transition happens from s_1 to initial state, as shown in Fig.1(a). If further confirmation is not needed, we set $\mu(s_1)$=0, and transition happens without delay. If needed, $e(\neg c_1)$ can be regarded as abnormal event pattern in this production stage, as shown in Fig.1(b).

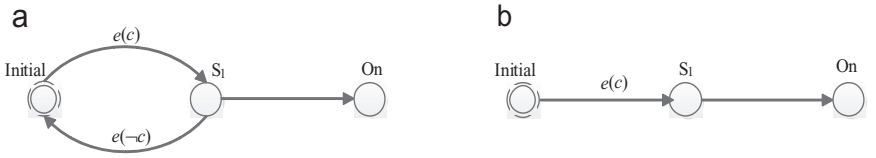

Fig. 1. (a) state transition with delay for atomic starting context; (b)state transition without delay for atomic starting context.

For some stages, there may be one more starting state, such as example in Fig.2, where c_1 and c_2 are two starting contexts, and $e(c_2)$ should happen after $e(c_1)$.

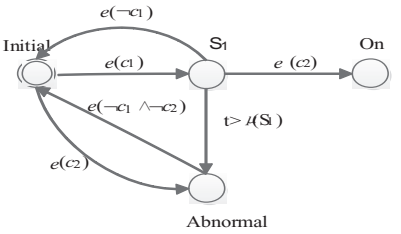

Fig. 2. state transition with two sequential atomic starting contexts

Fig.3 presents an example that $c_1 \wedge c_2$ is the start context of the stage. There may be two possible situations: $e(c_2)$ happens after $e(c_1)$, and $e(c_1)$ happens after $e(c_2)$. Fig.3(a) shows a direct way to represent the process. If we use event type with composite context, the representation can be simplified, as in Fig.3(b). Note that $e(c_1 \wedge c_2)$ can't be replaced by $e(c_1) \wedge e(c_2)$, because $e(c_1 \wedge c_2)$ is actually one basic event $e(c_1)$ or $e(c_2)$, different from the one happening before, while $e(c_1) \wedge e(c_2)$ are two basic events which happens in any sequence. Since one of $e(c_1) \wedge e(c_2)$ should happen before the transition from initial state to s_1, $e(c_1) \wedge e(c_2)$ here is not correct.

4. Case study

According to the approach presented above, we design state transition model of the stage in metallographic examination, and create abnormal event patterns in every stage.

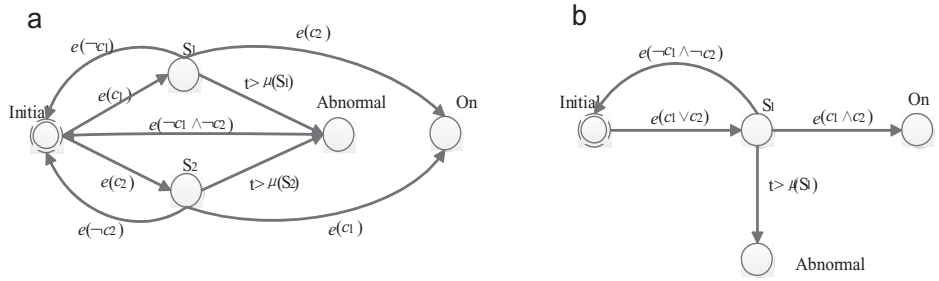

Fig.3. (a) state transition with two atomic starting contexts; (b) equivalent state transition with two atomic starting contexts

4.1. Architecture

Fig.4 presents the architecture of data flow in multi-stage abnormal situation monitoring framework. At the lowest level, raw readings from various sensing devices of the production site are collected and then filtered by the middleware to remove noisy data and transform to standard context or event format. The data are passed on to the complex event processing modular, consisted of complex event detection, adapter and database, for further processing.

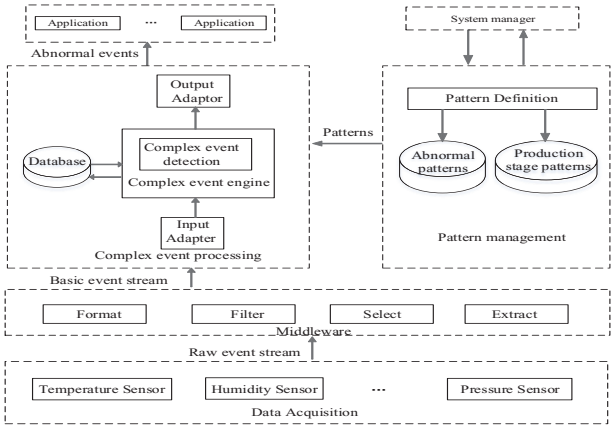

Fig.4 data flow in multi-stage monitoring system

The input adapter transforms the events into a fixed format for complex event detection. The output adapter passes the identified complex events in a user-defined form to application. Database stores historical events and the identified events. Here Esper, the CEP engine is used to match the event stream with patterns in the pattern library, if successfully, the engine will generate high-level complex events sequences.

Pattern library stores abnormal event patterns in every stage, starting context patterns and ending context patterns in stage transition period. Pattern definition converts the customized information into patterns that the engine can be parsed.

In this work, we implement the abnormal event identification in Java programming language and the Esper engine is used. Experiment has been done to test our proposed model.

4.2. Modeling abnormal situation monitoring in metallographic examination

Metallographic examination comprises several stages: sampling, inlaying, grinding, polishing and etching. According to possible abnormal event that may occur in every stage, we define abnormal event patterns. For every stage, we build state transition models. The most of stages is relatively simple which has only one starting context and one ending context except sampling stage.

In sampling stage, the worker accesses the lab, turns on the cutting machine, the wire winding motor and the pump motor. Then the worker adjusts the temperature and humidity to the specified value, puts the specimen in the specified position, and starts to operate. When the operation has been completed, the three machines should be turned off to end the sampling stage. Before the stage is ended, the three machines should be kept on, the temperature and the humidity should be in specified value.

Therefore the starting context is the three machines have been switched on and the ending context is the three switches are off. Fig.5 shows the state transition model of sampling stage.

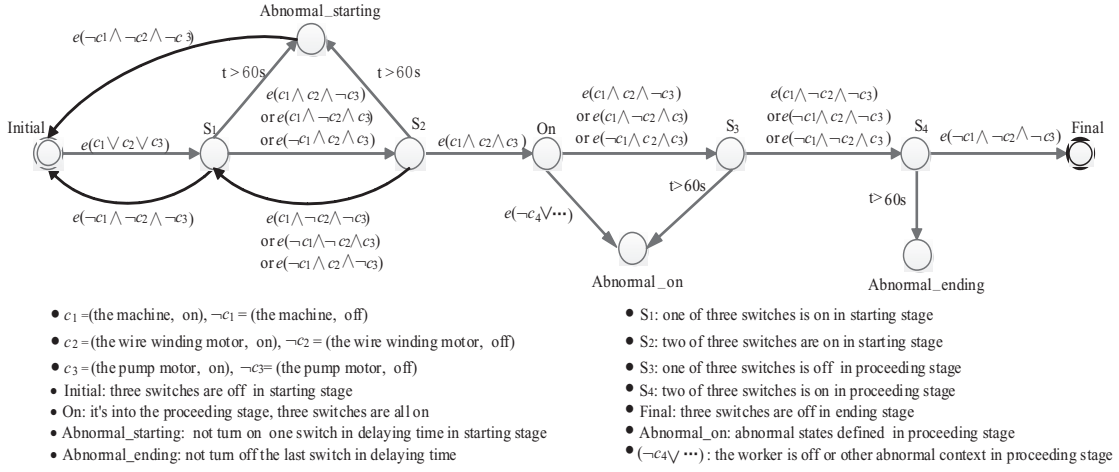

Fig. 5. stage transition model containing three switches

4.3 Experiment for abnormal situation detection

The experiment for abnormal event identification is implemented in Java and the Esper engine is used. Since only event detection part has been implemented, we construct experiment data sets with timestamp event streams simulating events input. The data is stored in database. The patterns have been predefined.

The main point of the experiment is to check: (1) all of the abnormal situations are detected; and (2) none of the normal situations are sorted into abnormal ones. We create groups of data, trying to include all of possibilities. Since sampling stage is relatively complex in the beginning and end, more experiments are performed in this stage. For example, Table 1 shows three groups data in sampling stage. There exist abnormal situation in these data such as the wire winding motor not being switched on in time and the pump being turned off during operations.

Table 1. part of experiment data in sampling stage

Timestamp	Sensor_ID	State	Timestamp	Sensor_ID	State	Timestamp	Sensor_ID	State
150	T001	20	210000	P001	On	806000	P001	On
200	H001	85%	220000	HP001	On	810000	HP001	On
310	U001	In	232000	W001	On	820800	W001	On
1400	W001	Off	240000	W001	Off	830000	HP001	Off
2600	P001	Off	310000	HP001	On	836000	W001	Off
3500	HP001	Off	320000	P001	On	910000	P001	On
10000	W001	On	330000	P001	On	920000	U001	Out
25000	P001	On	335000	U001	In	925000	U001	In
97000	HP001	Off	350000	P001	On	950000	P001	Off
…	…	…	…	…	…	…	…	…

By analyzing experiment results, we correct some of the event expressions which are not accurate in pattern library. The final experiment result shows that stage transition can be identified, all the abnormalities can be detected and none of the normal situation is regarded as abnormal one.

5. Conclusion

This paper focuses on the monitoring of abnormal situation in workspace where complicate production activities are performed and possible abnormal situations vary in different stages. Complex event processing is used to process high frequent data stream, tracking production process to identify production stage according to starting or ending context and detecting abnormal situation as user defined.

We present context model and event model in which discrete event such as acquiring context value at a point of time is represented by context. Contexts in different time captured by the sensors form event streams and can be processed by CEP engine. Abnormal situations that may happen in every stage are transformed into abnormal event patterns in event computing language supported by the selected CEP engine. For every stage there is a corresponding abnormal event pattern set.

For every production stage, we set starting context and ending context to mark the beginning and end of the stage. Since there are short transition periods at the beginning and in the end of the stage, where abnormal event patterns in the stage do not work, it is needed to distinguish normal transition from abnormal start and abnormal situations. We propose to use state transition to model the process.

In case study, we take metallographic examination into account to construct model and prove our approach. Results show that the approach is effective and feasible for some multi-stage abnormal situation monitoring.

The limitation of our approach is that in context composition we only consider conjunction, disjunction and negative operation which are far from enough to represent complex context relations. In addition uncertainty has not been considered in state transition model which exists in many real world problems. These problems should be taken into account in future research work.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant No. 71271038, 71531002).

References

[1] Oborski P. Developments in integration of advanced monitoring systems. *The International Journal of Advanced Manufacturing Technology*, 2014, 75(9-12): 1613-1632.

[2] Voisard A, Ziekow H. Modeling trade-offs in the design of sensor-based event processing infrastructures. *Information Systems Frontiers*, 2012, 14(2): 317-330.

[3] Seng WL. Representing and reasoning with situations for context-aware pervasive computing: a logic programming perspective. *The Knowledge Engineering Review*. 2005, 19(3): 213–233.

[4] Cugola G, Margara A. Complex event processing with T-REX. *Journal of Systems and Software*, 2012, 85(8):1709-1728.

[5] Luckham D. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems.* Addison-Wesley Longman Publishing Co. Inc. 2001:3-3.

[6] Yao W, Chu CH, Li Z. Leveraging complex event processing for smart hospitals using RFID. *Journal of Network and Computer Applications*, 2011, 34(3): 799-810.

[7] Wang F, Liu S, Liu P, et al. Bridging physical and virtual worlds: complex event processing for RFID data streams. *Advances in Database Technology-EDBT 2006*. Springer Berlin Heidelberg, 2006: 588-607.

[8] Wang F, Liu P. Temporal management of RFID data. *Proceedings of the 31st international conference on Very large data bases*. VLDB Endowment, 2005: 1128-1139.

[9] Huang Y, Zhang L, Xiang Q. RFID integrated real-time manufacturing monitoring based on complex event processing. *Journal of Tsinghua University (Science and Technology)*, 2013, 5: 024.

[10] Wang YH, Cao K, Zhang XM. Complex event processing over distributed probabilistic event streams. *Computers & Mathematics with Applications*, 2013, 66(10): 1808-1821.

[11] Jin B, Zhuo W, Hu J, et al. Specifying and detecting spatio-temporal events in the internet of things. *Decision Support Systems*, 2013, 55(1): 256-269.

[12] Dey AK, Abowd GD, Salber D. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 2001, 16(2-4): 97-166.

[13] Esper Team, EsperTech Inc. Esper Reference. http://www.espertech.com/esper/release-5.4.0/esper-reference/pdf/esper_reference.pdf.

[14] Byg J, Jacobsen M, Jacobsen L, et al. TCTL-preserving translations from timed-arc Petri nets to networks of timed automata. *Theoretical Computer Science*, 2014, 537: 3-28.
[15] Berthomieu B, Peres F, Vernadat F. Bridging the gap between timed automata and bounded time petri nets. *Formal Modeling and Analysis of Timed Systems*. Springer Berlin Heidelberg, 2006: 82-97.
[16] Adaikkalavan R, Chakravarthy S. SnoopIB: Interval-based event specification and detection for active databases. *Data & Knowledge Engineering*, 2006, 59(1): 139-165.
[17] Zhou C, Meng X, Chen Y. Out-of-order durable event processing in integrated wireless networks. *Pervasive and Mobile Computing*, 2011, 7(5): 595-610.
[18] Terroso-Saenz F, Valdes-Vela M, Skarmeta-Gomez A F. A complex event processing approach to detect abnormal behaviours in the marine environment. *Information Systems Frontiers*, 2015: 1-16.
[19] Coffi J R, Marsala C, Museux N. Adaptive complex event processing for harmful situation detection. *Evolving Systems*, 2012, 3(3): 167-177.
[20] Hinze A, Voisard A. EVA: An event algebra supporting complex event specification. *Information Systems*, 2015, 48:1–25.
[21] Obeid N, Rao R B K N. On integrating event definition and event detection. *Knowledge and information systems*, 2010, 22(2): 129-158.
[22] Ru Y, Cabasino M P, Giua A, et al. Supervisor synthesis for discrete event systems under partial observation and arbitrary forbidden state specifications. *Discrete Event Dynamic Systems*, 2014, 24(3): 275-307.
[23] Uraikul V, Chan C W, Tontiwachwuthikul P. Artificial intelligence for monitoring and supervisory control of process systems. *Engineering Applications of Artificial Intelligence*, 2007, 20(2): 115-131.
[24] Lin W C, Garcia H E, Yoo T S. A diagnoser algorithm for anomaly detection in DEDS under partial and unreliable observations: characterization and inclusion in sensor configuration optimization. *Discrete Event Dynamic Systems*, 2013, 23(1): 61-91.
[25] Bruns R, Dunkel J, Masbruch H, et al. Intelligent M2M: Complex event processing for machine-to-machine communication. *Expert Systems with Applications*, 2015, 42(3):1235-1246.